This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method of generating a vendor-provided manifest that governs the execution of a software object distributed by the vendor, the method comprising:

creating a description that specifies requirements that are to be embodied in the vendor-provided manifest; and

~~receiving a specification indicative of requirements for the execution of the software object, the specification referring to one or more components;~~

providing the description to a manifest generation tool that reads the description and generates the vendor-provided manifest based on the requirements, ~~generating a manifest based on said specification, including accessing said one or more components,~~ said manifest comprising one or more rules imposed by the vendor for ensuring integrity of an address space that is used in a computer for executing the software object, ~~for execution of the software object,~~ the one or more rules incorporating a list of acceptable and unacceptable modules, wherein the acceptable modules may be executed in the address space of the computer and the unacceptable modules are unconditionally barred from being executed in the address space of the computer.

2. (Currently Amended) The method of claim 1, wherein said ~~specification~~ description identifies the acceptable and unacceptable modules, and wherein generating the manifest comprises including, in said manifest, the identities of the ~~the~~ acceptable and unacceptable modules identified in the ~~specification~~ description.

3-5. (Canceled)

6. (Currently Amended) The method of claim 2, wherein said ~~specification~~ description indicates whether said manifest will contain hashes for identifying the unacceptable modules.

7. (Currently Amended) The method of claim 1, wherein at least one of said acceptable modules comprises a key, and wherein said ~~specification~~ description indicates that the at least one of said acceptable modules signed with said key may be loaded into said address space, and wherein generating said manifest comprises:

retrieving said key from a file identified in said ~~specification~~ description; and

including said key in said manifest.

8. (Previously presented) The method of claim 1, wherein generating said manifest comprises:

computing a hash of at least one of said unacceptable modules; and

including said hash in said manifest.

9. (Currently amended) The method of claim 1, wherein said generating act comprises:

based on said ~~specification~~ description, creating a data structure representative of said ~~specification~~ description; and

generating said manifest based on said data structure.

10. (Previously presented) The method of claim 1, further comprising:

receiving a key associated with a vendor or distributor of said software object;

signing said manifest with said key to produce a digital signature; and

including said digital signature in said manifest.

11. (Original) The method of claim 1, further comprising:

using a hardware security module to sign said manifest, said hardware security module being adapted to apply a key associated with a vendor or distributor of said software object without revealing said key outside said hardware security module.

12. (Currently Amended) A computer-readable medium encoded with computer-executable instructions to perform a method of generating a manifest that governs the execution of a

software object distributed by a vendor, the method comprising:

parsing a specification of requirements to be included in the manifest, the requirements ~~defining~~ comprising a vendor-specified policy configured to preclude loading of a rogue module into an address space of a computer in which the software object is to be executed; ~~of a software object associated with the manifest;~~

accessing one or more components that are identified by the specification and that are external to the specification, the one or more components including an executable module; and

generating a manifest based on at least one of the accessed objects, the generation comprising:

including in said manifest an identification of said executable module and an indication that either:

said executable module may be loaded into said address space; or

said executable module may not be loaded into said address space.

13. (Canceled)

14. (Previously presented) The computer-readable medium of claim 12, wherein said rogue module is operative to perform an unauthorized operation on the one or more components.

15-16. (Canceled)

17. (Currently Amended) A method of specifying constraints on the use of a software object, the method comprising:

creating a specification ~~for explicitly limiting~~ that permits a vendor to specify what may be loaded into an address space of ~~the software~~ a computer in which the software object is to be executed, the specification referring to one or more components that are external to the software and external to the specification;

using a manifest generation tool to generate a manifest based on the specification,

wherein the manifest generation tool does at least one of:

including, in said manifest, data from one of said one or more components; or

computing a value based on one of said one or more components and including the computed value in said manifest; and

distributing the generated manifest together with the software object, ~~wherein the manifest comprises rules explicitly limiting what may be loaded into the address space of the software,~~ thereby ensuring a secure address space for executing the software object.

18. (Original) The method of claim 17, wherein said one or more components comprises a module, wherein said specification indicates either that said module may be loaded into said address space or that said module may not be loaded into said address space, and wherein said manifest generation tool does at least one of:

including an identifier of said module in said manifest; or

computing a hash of said module and including the hash in said manifest.

19. (Original) The method of claim 17, wherein said one or more components comprise a key, wherein said specification indicates either that modules signed with said key may be loaded into said address space or that modules signed with said key may not be loaded into said address space, and wherein said manifest generation tool retrieves said key from a file identified in said specification, and includes a certificate for said key in said manifest.

20. (Original) The method of claim 17, wherein said manifest generation tool creates an intermediate data structure representative of said specification, and generates said manifest based on said intermediate data structure.

21. (Original) The method of claim 17, wherein the method further comprises:

receiving a key from further comprising:

receiving a key associated with a vendor or distributor of the software;

signing said manifest with said to produce a digital signature; and

including said digital signature in said manifest.

22. (Original) The method of claim 17, further comprising:

using a hardware security module to sign said manifest, said hardware security module being adapted to apply a key associated with a vendor or distributor of the software without revealing said key outside said hardware security module.

23. (Currently amended) A system comprising a processor for generating a manifest, the system comprising:

a first parser implemented on the processor, the first parser configured to receive ~~that receives~~ a manifest specification indicative of requirements for a manifest, the first parser generating a representation of said requirements, said requirements relating to what may be loaded into an address space of a software object, said specification referring to one or more components external to said software and external to said specification; and

a first manifest generator that generates a manifest based on said representation and includes in said manifest information contained in, or computed based on, said one or more components, the manifest configured to interoperate with ~~; and~~ a security component that imposes a permeable barrier for selectively allowing acceptable modules to be loaded into the software space of the software object and blocking unacceptable modules from being loaded into the software space thereby preventing unauthorized tampering of the one or more components.

24. (Original) The system of claim 23, wherein said one or more components comprise a module, and wherein said first manifest generator generates said manifest by including, in said manifest, a datum that identifies said module.

25. (Previously Presented) The system of claim 24, wherein said datum comprises a hash of said module.

26. (Previously presented) The system of claim 23, wherein said one or more components comprise a key, wherein said specification indicates either that acceptable modules signed with said key may be loaded into said address space or that unacceptable modules signed with said key may not be loaded into said address space, and wherein said first manifest generator retrieves said key from a file identified in said specification and includes said key in said manifest.

27. (Original) The system of claim 23, wherein said first manifest generator generates a digital signature for said manifest by signing said manifest with a key associated with a vendor or distributor of said software object, and includes said digital signature in said manifest.

28. (Canceled)

29. (Original) The system of claim 23, further comprising:

a second parser that receives a manifest specification indicative of requirements for a manifest, the second parser generating a representation of said requirements in the same format as said first parser,

wherein said first parser parses specifications in a first format and second parser parses specifications in a second format different from said first format, and wherein first manifest generator generates said manifest based on a representation produced either by said first parser or said second parser.

30. (Original) The system of claim 23, further comprising:

a second manifest generator that generates a manifest based on said representation, wherein said first manifest generator generates a manifest in a first format and second manifest generator generates a manifest in a second format different from said first format.

31. (Previously presented) The method of claim 1, wherein at least one of the unacceptable modules is identified in the list by a version number.

32. (Previously presented) The method of claim 1, wherein at least one of the unacceptable modules is identified in the list by a range of version numbers.

33. (Previously presented) The computer-readable medium of claim 12, wherein the policy comprises an identity of an unacceptable module that is unconditionally barred from being executed in the address space of the software object.

34. (Previously presented) The computer-readable medium of claim 33, wherein the unacceptable module is identified in the policy by a hash identifier.